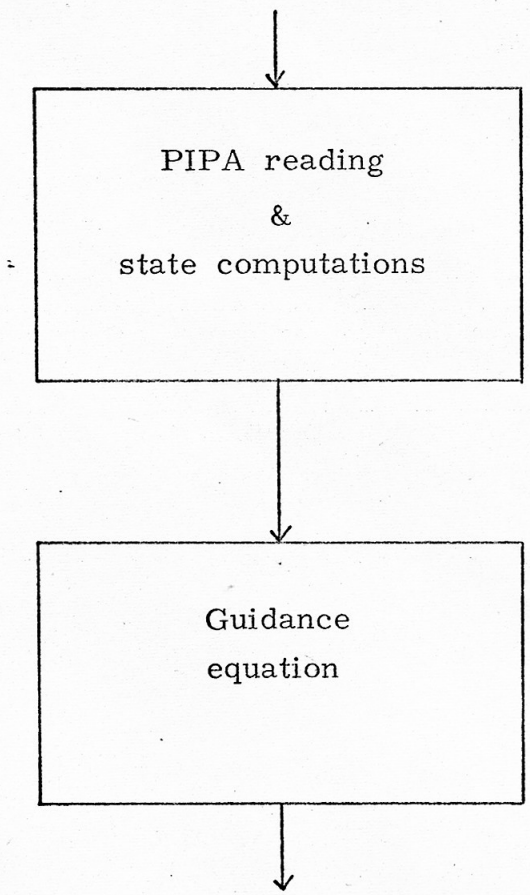
8-5-70

DEFINITION AND DESCRIPTION OF THE VARIABLE GUIDANCE PERIOD SERVICER

Eyles

What is Servicer?

Servicer is a routine which runs periodically during powered flight in which navigation and guidance are performed. Servicer begins with the reading of the accelerometers, the average-G equation, and other vehicle state computations. This part is common to all the powered flight phases. Servicer next transfers control to the appropriate guidance equation and concludes after the output of displays and of commands to the autopilot and throttle.



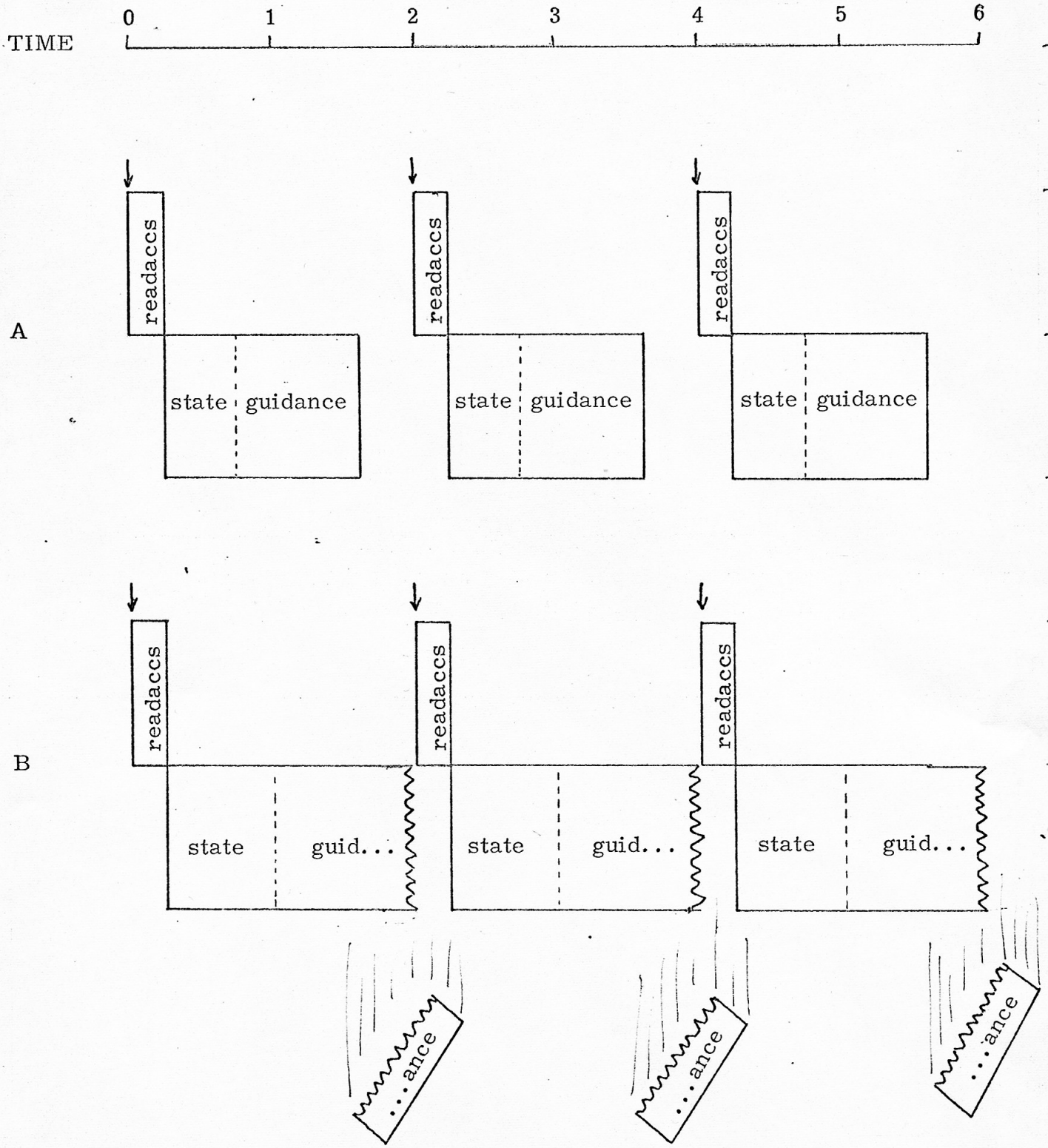
Fixed and Variable Servicers

If the reading of the PIPAs occurs at a constant frequency the Servicer is said to be fixed. Guidance period does not vary. If, on the other hand, the time of reading the PIPAs can vary in response to variations in the length of the Servicer computations, the Servicer is variable. In this case guidance period changes and must be computed instead of being assumed.

LUMINARY's Servicer

The Servicer now in LUMINARY is a fixed Servicer which runs with a period of 2 seconds. It is started every pass by an interrupt — the READACCS task — in which the PIPAs are read and a Servicer job set up. This job is set up whether or not the job initiated by the previous READACCS task has been completed. When TLOSS or extra computations cause the Servicer job to expand to the extent that it is not completed before the next READACCS, the next Servicer job begins anyway. The remainder of the first Servicer job is stored in the core sets and vac areas to be executed later, whenever there is time. This means (1) that the work at the end of the Servicer job does not get done, (2) that dangerous and very complex bahavior may occur when odd left-over pieces of Servicer jobs at last get executed, and (3) that eventually software restarts result when core sets (alarm 31202) or vac areas (alarm 31201) are exhausted.

Here are pictures of the LUMINARY Servicer, in normal operation (A), and in trouble thanks to TLOSS (B).



ZERLINA History

The Servicer developed in ZERLINA is a second-generation variable Servicer. The first was developed over a year ago in another off-line version, DIANA by MOONLIGHT. Here guidance period was not continuously variable, but it could be changed by the astronaut in increments of 1/4 second by means of extended verbs. Despite its shortcomings this version contributes to the present effort (1) an idea of the difficulties and the magnitude of the effort involved in incorporating a variable Servicer, and (2) some specific blocks of coding, notably the average-G equations. Landings and the P40s were tested in this early version.

ZERLINA by ZOROASTER originated as a version of LUMINARY revision 145 in February of this year. By revision 18 it contained a variable Servicer essentially complete and tested in all the powered-flight phases of the nominal mission. In later revisions of ZERLINA a new Landing Analog Displays routine was developed — which has already found its way into LUMINARY. Throughout, ZERLINA has been kept up to date with the main-line LUMINARY and ZERLINA revision 35 is the equivalent of LUMINARY revision 174 except for its superior Servicer. ZERLINA shares most of its subroutines with LUMINARY.

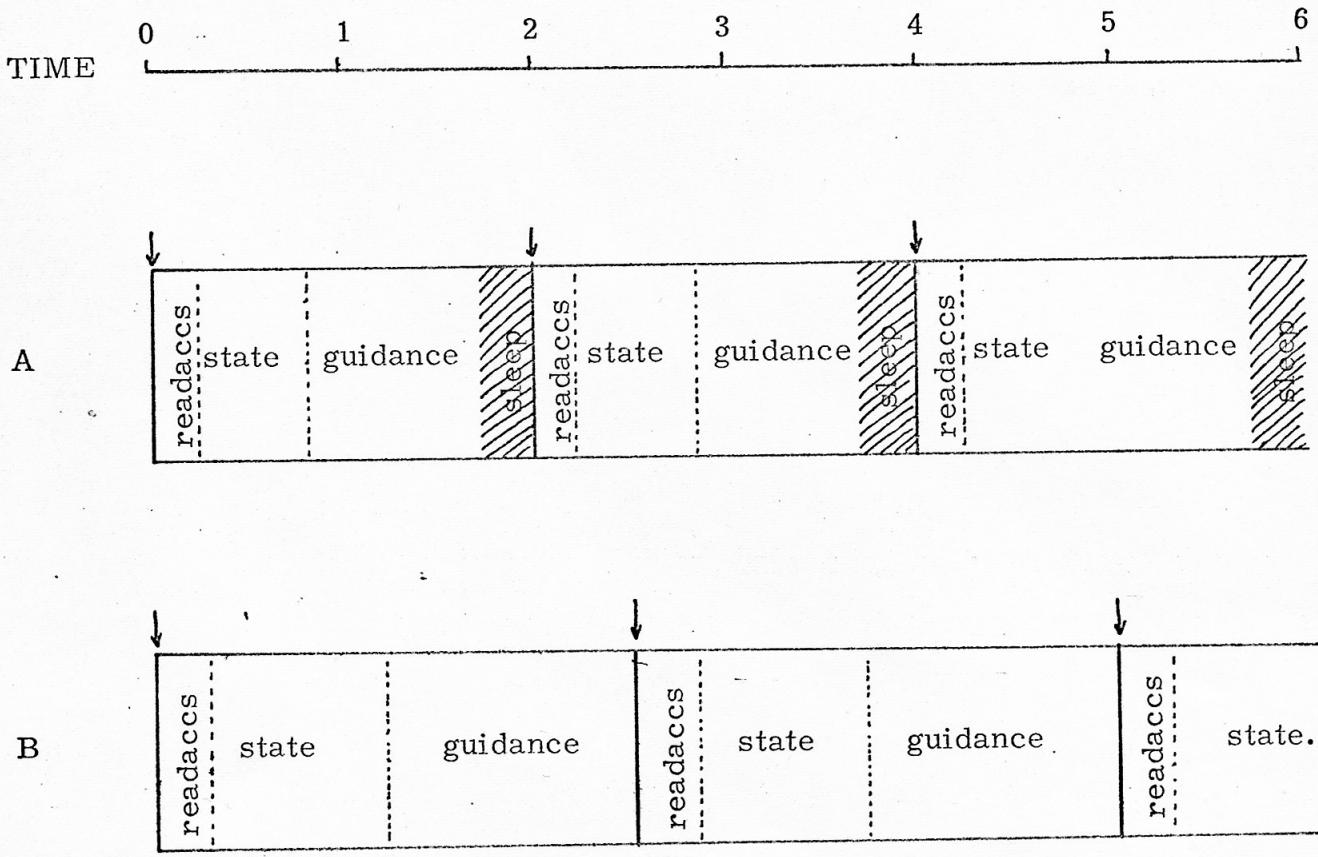
ZERLINA's Servicer

The variable Servicer developed in ZERLINA runs with a minimum period of 2 seconds. In normal operation when its computations are completed the Servicer job sleeps until the 2 seconds are up. But if TLOSS or extra work expand the Servicer duty cycle to the extent that it takes more than 2 seconds, the start of the suceeding Servicer cycle is slipped until it finishes. Guidance period is automatically stretched to fit the required duties and sleep time is eliminated. Because in this case the Servicer job is always active — at priority 20 — the other jobs that must be executed are given higher priorities to enable them to break in.

Since it can vary, guidance period is computed every pass. This is turn frees Servicer from rigid timing constraints and so the READACCS task can be eliminated. The PIPAs are read in the Servicer job. It makes restart protection easier and increases Servicer's ruggedness under time-stress.

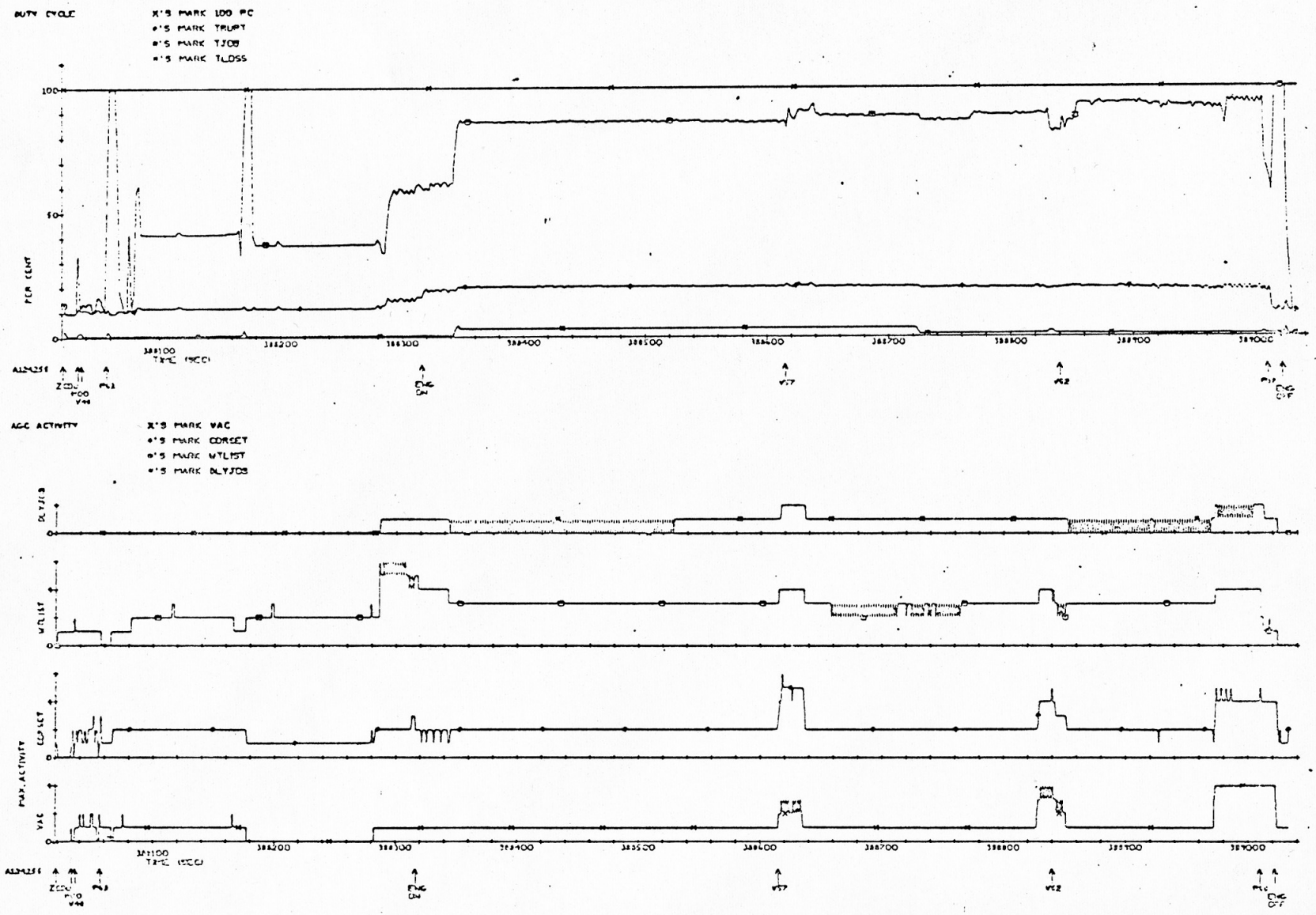
In ZERLINA the Servicer job runs end-to-end and thus cannot overlap itself. When it finishes it simply starts over. This means that the guidance equation — whichever is connected — must after issuing its commands transfer control to the beginning of Servicer, marked by the tag PIPCYCLE, instead of ending the job.

Here are pictures of the ZERLINA Servicer, in normal operation (A), and adjusting to high TLOSS (B).



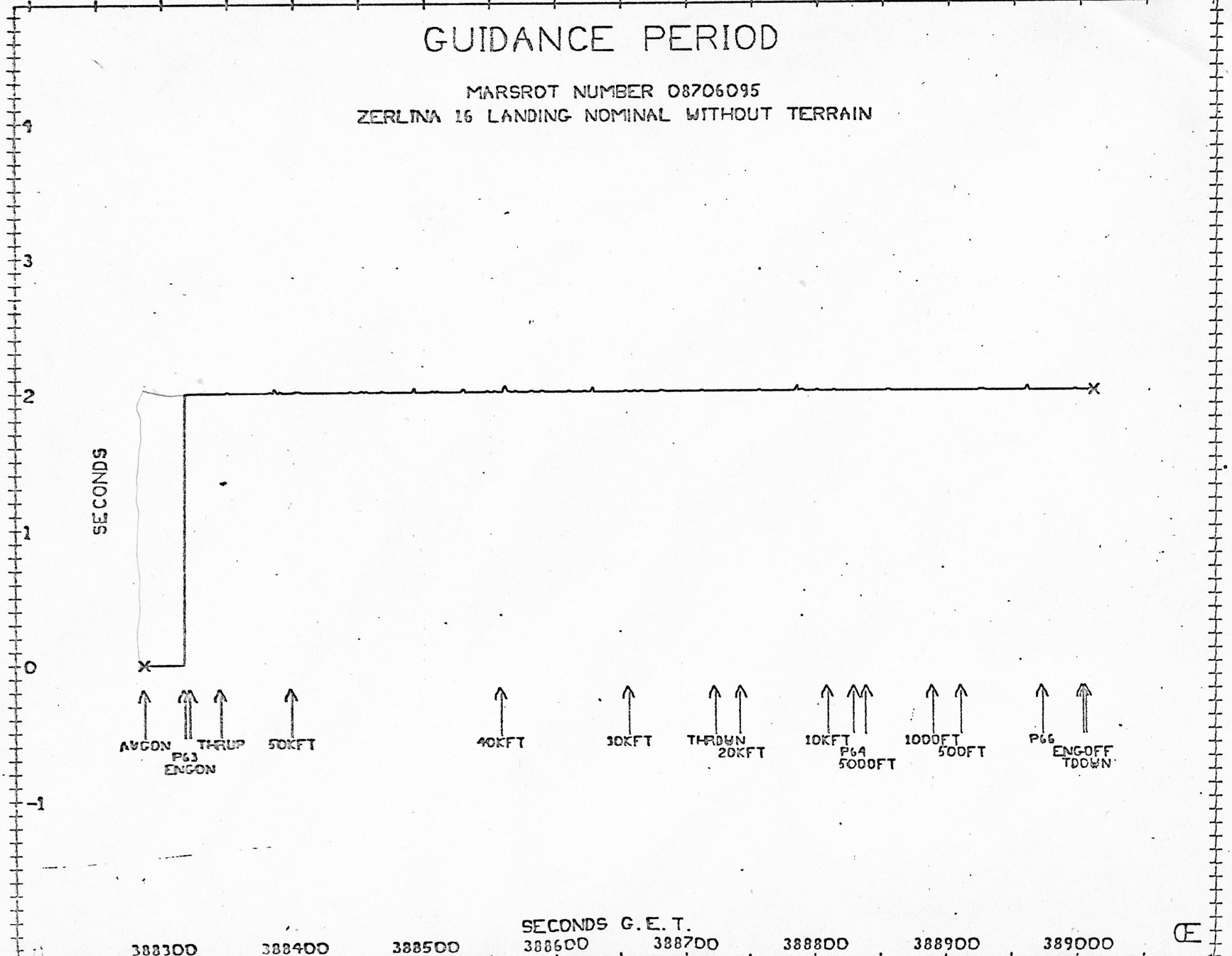
On the next pages are dutycycle and guidance period plots for Landings run on ZERLINA with zero and with 20% TLOSS. After that comes a flowchart of the Servicer job and its auxilliaries.

ZERLINA LANDING: NO TLOSS

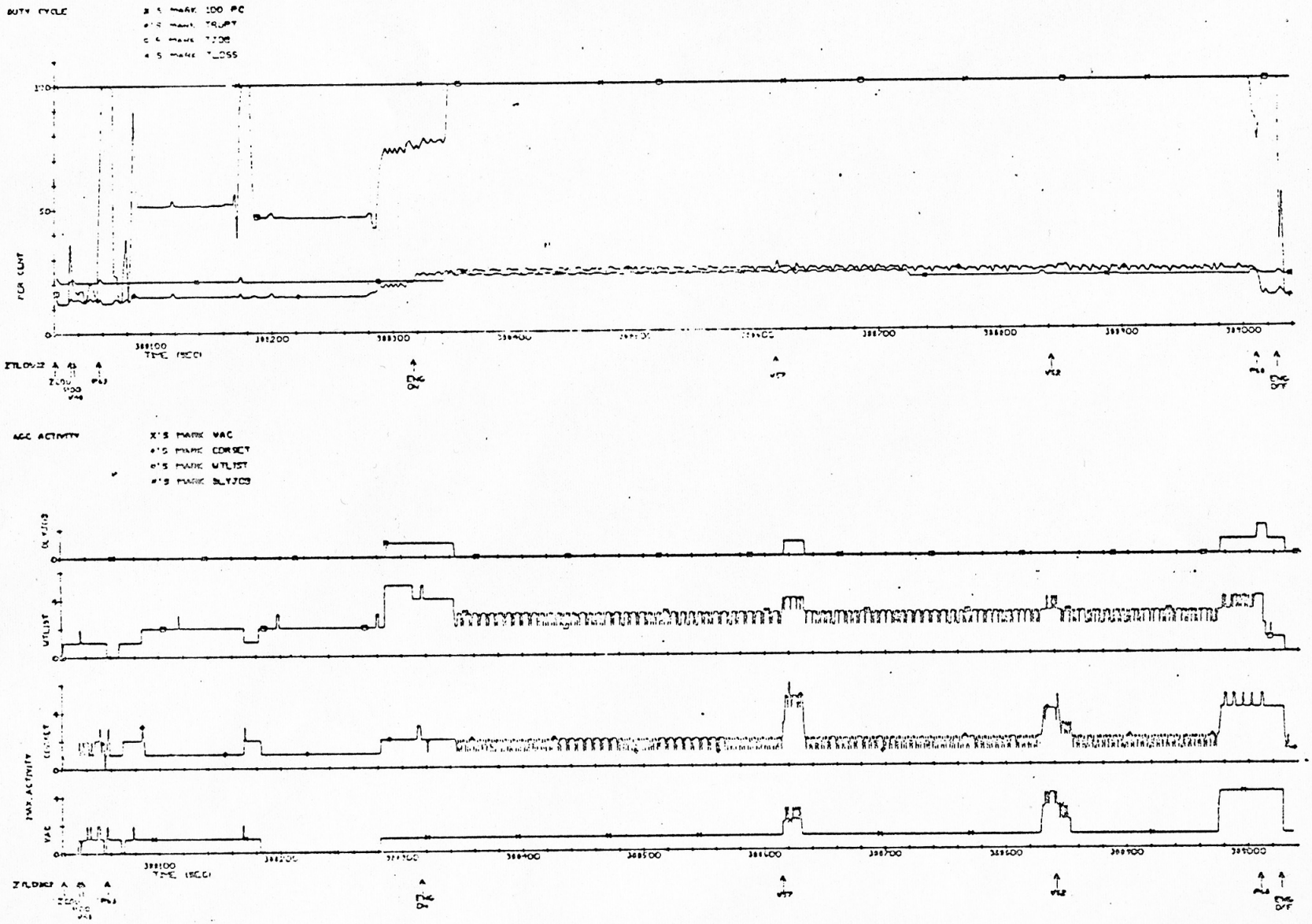


GUIDANCE PERIOD

MARSROT NUMBER 08706095
ZERLINA 16 LANDING NOMINAL WITHOUT TERRAIN

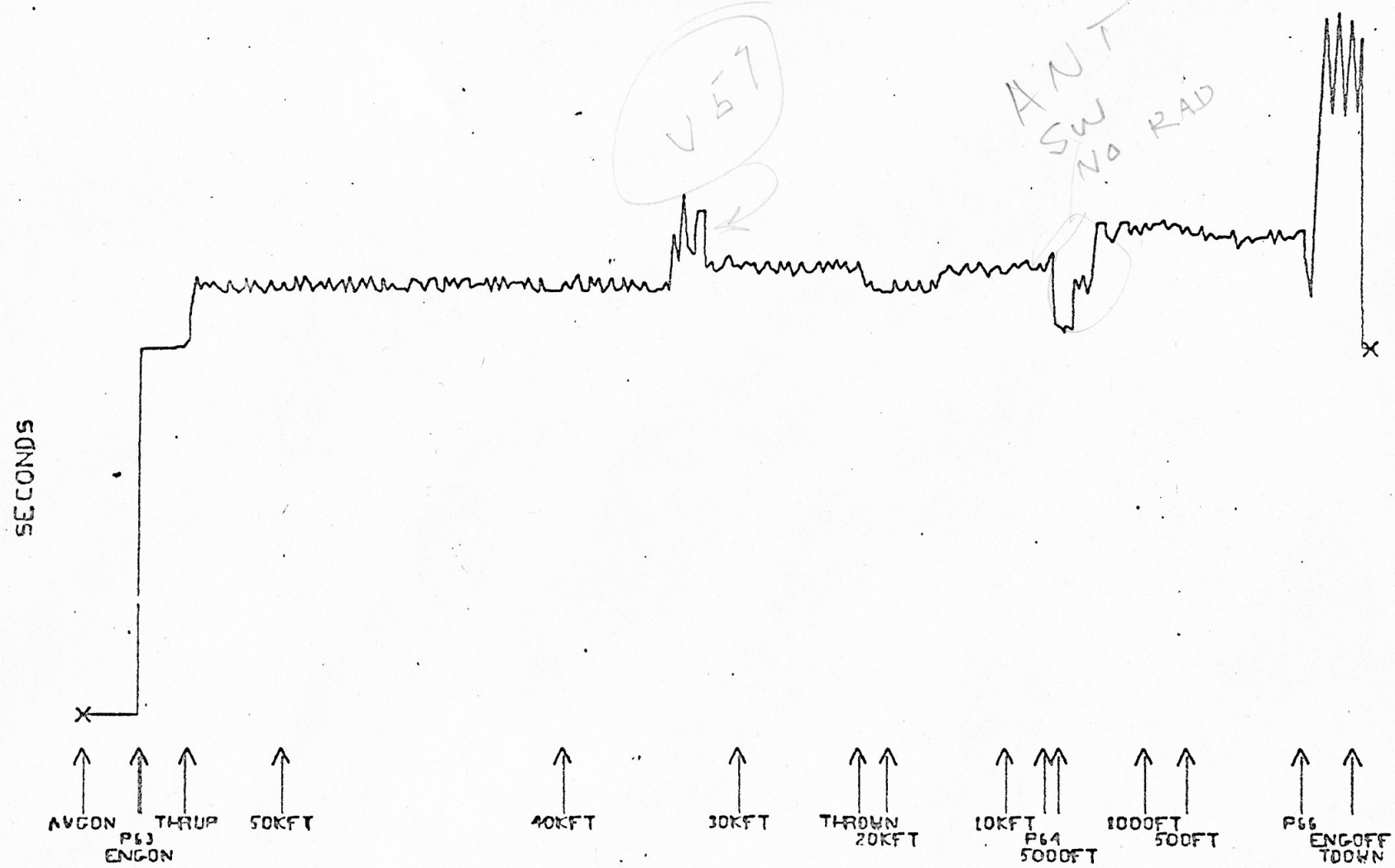


ZERLINA LANDING: 20% TLOSS



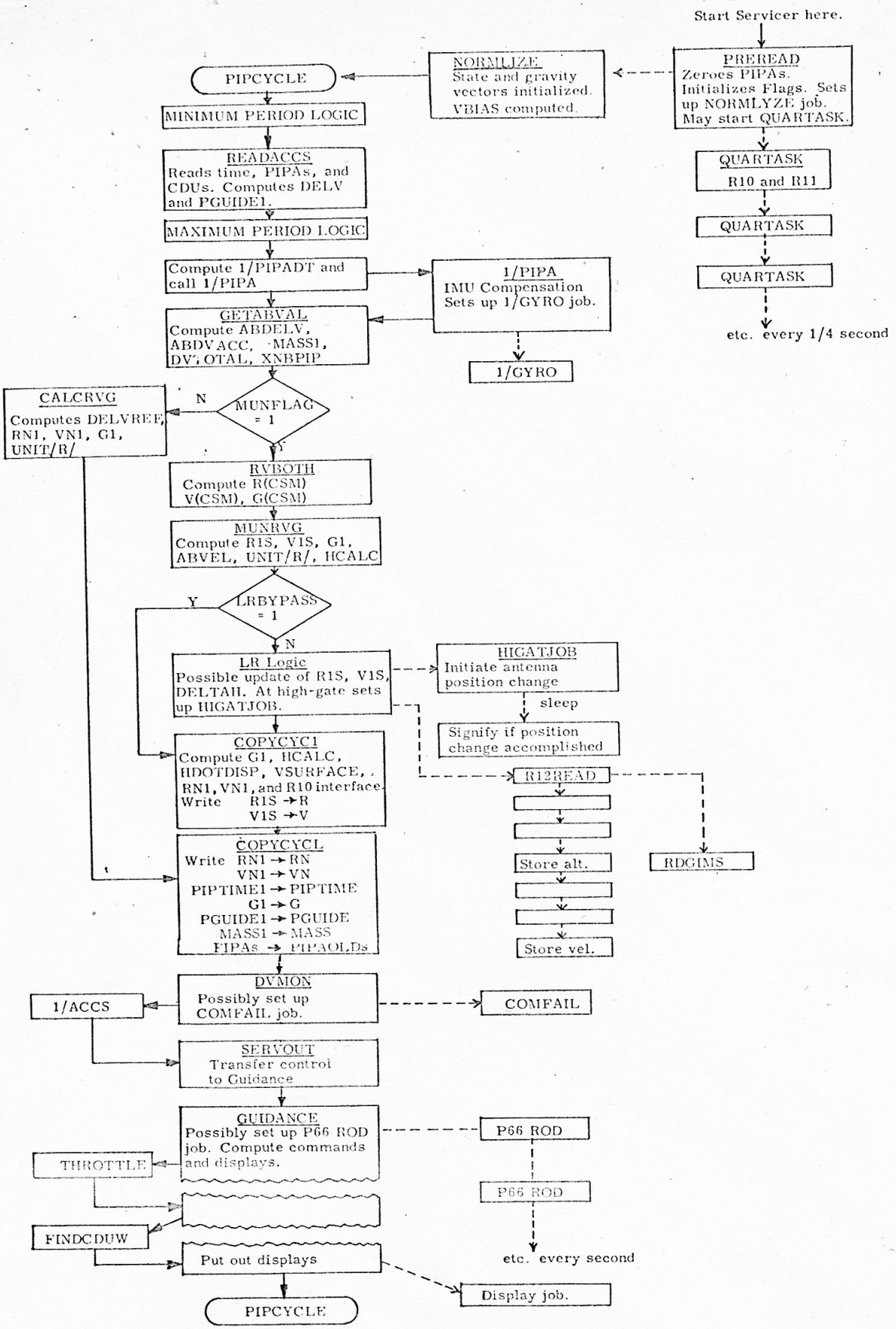
GUIDANCE PERIOD

MARSROT NUMBER 08720472
ZERLINA 16 LANDING WITH 20 PERCENT TLOSS



SECONDS G.E.T.

388300 388400 388500 388600 388700 388800 388900 389000


Start Servicer here.
PREREAD
Zeroes PIPAs. Initializes Flags. Sets up NORMLYZE job. May start QUARTASK.
NORMLIZE
State and gravity vectors initialized. VBIAS computed.
PIPCYCLE
QUARTASK
R10 and R11
QUARTASK
QUARTASK
etc. every 1/4 second
MINIMUM PERIOD LOGIC
READACCS
Reads time, PIPAs, and CDUs. Computes DELV and PGUIDE1.
MAXIMUM PERIOD LOGIC
Compute 1/PIPADT and call 1/PIPA
1/PIPA
IMU Compensation Sets up 1/GYRO job.
GETABVAL
Compute ABDELV, ABDVACC, MASS1, DVTOTAL, XNBPIP
1/GYRO
CALCRVG
Computes DELVREE, RN1, VN1, G1, UNIT/R/
N
MUNFLAG = 1
Y
RVBOTH
Compute R(CSM) V(CSM), G(CSM)
MUNRVG
Compute R1S, V1S, G1, ABVEL, UNIT/R/, HCALC
Y
LRBYPASS = 1
N
LR Logic
Possible update of R1S, V1S, DELTAH. At high-gate sets up HIGATJOB.
HIGATJOB
Initiate antenna position change
sleep
Signify if position change accomplished
COPYCYC1
Compute G1, HCALC, HDOTDISP, VSURFACE, RN1, VN1, and R10 interface.
Write R1S → R
V1S → V
R12READ
Store alt.
Store vel.
RDGIMS
COPYCYCL
Write RN1 → RN
VN1 → VN
PIPTIME1 → PIPTIME
G1 → G
PGUIDE1 → PGUIDE
MASS1 → MASS
PIPAs → PIPAOLDs
DVMON
Possibly set up COMFAIL job.
1/ACCS
COMFAIL
SERVOUT
Transfer control to Guidance
GUIDANCE
Possibly set up P66 ROD job. Compute commands and displays.
P66 ROD
P66 ROD
etc. every second
THROTTLE
FINDCDUW
Put out displays
Display job.
PIPCYCLE

Variable Servicer Topics

PREREAD is changed in one way. Formerly the task in which the Landing Analog Displays, the Abort Monitor, and certain Landing Radar functions are performed (called QUARTASK in ZERLINA) was set up for 8 passes by the READACCS task. Now it is set up for an indefinite number of passes by PREREAD.

Minimum Period Logic. First to be executed after PIPCYCLE is the Minimum Period Logic: if it has been less than PGMIN since the last PIPA reading then Servicer goes to sleep for the remaining time. PGMIN is 2 seconds. The reasons for such logic are (1) to establish a (hard) lower bound on guidance period for scaling purposes, (2) to provide a mechanism for granting (but not guaranteeing) some time to low priority extended verbs, and (3) to insure enough time for at least one full downlist to be sent for each Servicer cycle. Picking PGMIN to be 2 seconds has the additional advantage of making it unnecessary to remove the assumption of a 2 second period from the P40s guidance equations. The margin during these phases is so great that even the TLOSS at which the DAP breaks down is insufficient to make PGUIDE exceed 2 seconds.

Cyclical PIPA Reader is the name of one of the devices which made the writing of this variable Servicer somewhat easier. This works as follows: after PREREAD the PIPAs are never zeroed and each DELV is computed, not read from the PIPA, as the difference between this and the previous value of the PIPA, stored in PIPAOLD. Correction for the chance that the PIPA in question overflowed

between the two readings is easily performed under the comfortable assumption that 81.92 m/s cannot be honestly accumulated during one PIPA interval. (With DPS thrusting full-bore and both DPS and APS tanks empty it would take over 8 seconds to build up this velocity.) Advanteges of this PIPA reading philosophy are (1) the straightforwardness it gives the PIPA reader restart protection, and (2) that it lets asynchronous routines like the P66 ROD equation work with a completely different PIPA interval without the interference of a periodic zeroing of the PIPAs by Servicer.

Maximum Period Logic. Here if PGUIDE1 exceeds PGMAX alarm 555 is issued. This is a plain alarm, not a POODOO or BAILOUT. This logic establishes a (soft) upper limit on guidance period for scaling purposes. 5 seconds is a reasonable value for PGMAX.

1/PIPA. Guidance period is input to the IMU compensation routine 1/PIPA (and 1/GYRO) through the register 1/PIPADT, which is computed every pass from PGUIDE1. The scaling of 1/PIPADT had to be changed from units of 2^8 centiseconds to units of 2^{10}.

ABDVACC. ABDELV has always contained delta-V magnitude over the PIPA interval. With the PIPA interval a constant 2 seconds ABDELV could be used as an acceleration, but with guidance period varying it cannot be. So it is supplemented by ABDVACC, a true acceleration computed from ABDELV by multiplying it by 2SECS/PGUIDE1 for use by the delta-V Monitor, the Ascent Guidance (in place of ABDVCONV), and 1/ACCS.

Average-G. The average-G equations were modified to take account of guidance period and to compute G, a gravitational acceleration in units of 2^{-6} m/cs/cs, instead of GDT/2, a velocity in which the DT is an implicit 2 seconds. G suits most users better than GDT/2. The average-G equations in LUMINARY are:

PGUIDE = 2 seconds

R1S = PGUIDE (V + DELV/2 + GDT/2) + R

Compute GDT1/2 from R1S.

V1S = GDT1/2 + DELV/2 + GDT/2 + DELV/2 + V

In ZERLINA these become:

PGUIDE1 = PIPTIME1 - PIPTIME

R1S = (PGUIDE1 G/2 + DELV/2 + V) PGUIDE1 + R

Compute G1 from R1S.

V1S = PGUIDE1 G1/2 + PGUIDE1 G/2 + DELV/2
+ DELV/2 + V

Landing Radar Incorporation Logic. Changes in the landing radar portion of Servicer include the addition of a state vector integration to find LM altitude and velocity at the time of the radar read. This is necessary because R12READ cannot be synchronized with PIPTIME, as it is in current LUMINARYs, because PIPTIME is not known in advance. Also a Nav Base to Stable Member transformation must be performed on the antenna beam vectors which is valid at the time of the radar read. RDGIMS, a task set up by R12READ , records the PIPAs, the CDUs, and time (in register LRTIME) at the mid-point of the read. Using consistent values for the time of the read Servicer computes a position and a velocity correction which are used to modify R1S and V1S. These changes are almost like putting back coding which existed in LUMINARY 1C and earlier when the velocity read was not synchronized. This is the most time consuming of the extra computations required by the variable Servicer, but it does not prevent Landings from running at the minimum period of 2 seconds throughout in the absense of TLOSS. During Landing Radar operation the CDUs and time for the radar read sent out on the downlist are the values recorded in RDGIMS, not the values at PIPTIME as in LUMINARY. Finally, HCALC1 is used for altitude at the time of the radar read. The register HCALC is used for DSKY display, in P66 as in the other phases. In LUMINARY, for no good reason, both are used for display.

Changes outside Servicer

Restart Tables. Servicer in ZERLINA has only a job to restart protect, the READACCS task having been eliminated, and so the special variable phase-change routine SERVCHNG and the type of PHASCHNG in which the restart point is specified by a 2CADR in-line are all that is needed in establishing restart points. Thus all the group 5 restart tables can be deleted.

Pinball Noun Tables. Because HCALC1 is only altitude at the time of the radar read and HCALC is at all times the altitude suitable for display on the DSKY, nouns 60, 63 and 92 should have HCALC substituted for HCALC1.

Landing Guidance Equations. The Landing guidance has to be modified to use G instead of GDT/2. Additionally, in the same section, priority is raised to 23 before the display routine is called and lowered again to 20 afterwards. This is so the off-line display job will have sufficient priority to break in on the Servicer job and post its displays even in the presense of TLOSS. Also, the VACRLEAS calls at the end of the Landing guidance are removed. Otherwise Servicer would start again at PIPCYCLE without a vac area.

P66 Guidance. Of the two P66 ROD computations performed every 2 seconds, in LUMINARY one is part of the Servicer job and the other is separate. In ZERLINA both are outside of the Servicer job in an asynchronous loop with a 1 second period. This loop is initiated and initialized when P66 is selected. The P66 ROD equation is executed every 1 second regardless of how slowly Servicer is running. The P66 horizontal control equation remains part of the Servicer job and will be executed less often when there is high TLOSS. In the P66 ROD equation G takes the place of GDT/2 and adjustments are made to adapt it to the Cyclical PIPA Reader.

Throttle Control Routine. Because a 2 second guidance period is implicit both in ABDELV and in FWEIGHT (a thrust compensation number, but really porportional to velocity not acceleration), both used in the computation of present thrust, this computation is changed. In LUMINARY it is

FP = ABDELV MASS + FWEIGHT.

In ZERLINA it becomes

FP = (ABDELV MASS + FWEIGHT) 2SECS/PGUIDE.

FINDCDUW wishes to complete the commanded attitude manoeuvre in exactly one guidance period. Thus, if guidance period expands, the rate commanded by FINDCDUW must be diminished. This is done by the usual method of multiplying by 2SECS/PGUIDE. Thus the guidance period from the preceeding full pass is used to predict the period for the pass about to begin.

When guidance period is fluctuating from pass to pass, as it sometimes does in P66, this prediction may be wrong and an attitude overshoot may result. However the time constant of the P66 horizontal equation (5 seconds) is sufficiently long to assure stability.

Ascent Guidance. In the Ascent Guidance routine G replaces GDT/2, ABDVACC replaces ABDVCONV as the input to the thrust magnitude filter, and, as in the Landing guidance, priority is raised before the display routine is called and lowered afterwards.

Landing Analog Displays are adapted to the Cyclical PIPA Reader. Also, since QUARTASK continues indefinitely once set up by PREREAD, there is no need to count passes in this routine.

Abort Interface. When a P70 or P71 abort is commanded a software restart is performed to flush the Landing guidance. In LUMINARY this guidance is protected in a different group than Servicer (group 3) and so the flushing is accomplished by turning off group 3 before the restart. In ZERLINA both Servicer and the guidance equation are protected in group 5. When Servicer transfers control to the guidance equation it sets the new SERVOVER flag; when control returns to Servicer SERVOVER is reset. Accordingly, before its software restart, the abort lead-in checks the SERVOVER flag and if it is set, indicating guidance in progress, the address of PIPCYCLE is substituted for the guidance address in the group 5 restart registers. Thus only Servicer, and not the guidance equation, is restarted after the software restart.

IMU Compensation. In 1/PIPA and 1/GYRO, the IMU compensation routines called by Servicer every pass, slight changes must be made to adapt to the new scaling of 1/PIPADT.

1/ACCS in LUMINARY uses ABDELV as an acceleration. In ZERLINA it is modified to use ABDVACC, a true acceleration, instead.

ZERLINA in Operation

The operational differences between ZERLINA and LUMINARY are in what will not happen, not in what will. The possibility of 31201 and 31202 alarms and anomalous behavior due to TLOSS is drastically diminished. V16 monitors can be used without fear in ZERLINA.

Alarm 1466, issued by LUMINARY when a P66 guidance pass is dropped, does not exist in ZERLINA. P66 guidance is never dropped. One new alarm does exist in ZERLINA, already mentioned in the Maximum Period Logic: alarm 555. With PGMAX set at 5 seconds this alarm will not occur with a TLOSS lower than 20%. If it did occur it would probably do so in P66 when guidance period is greatest. In this case rapid nulling of horizontal velocity could not be expected from the P66 horizontal equation and probably attitude should be controlled manually in attitude-hold. ROD performance will be unaffected because the P66 ROD equation continues to run every second.

Another alarm sometimes seen in very high TLOSS ZERLINA runs is alarm 32000. This alarm, with a software restart, is issued when the DAP cycle overlaps itself. It would appear in LUMINARY runs too if other things did not happen first. Testing so far indicates that the correct response to this alarm is to press on.

But I should end up by emphasizing that TLOSS will cause no alarm in ZERLINA until it reaches the neighborhood of 20%.